UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

(2)

## IMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. **AD-A220 498**

1b. RESTRICTIVE MARKINGS

2a.

3. DISTRIBUTION/AVAILABILITY OF REPORT
DISTRIBUTION STATEMENT A- Approved for public release; distribution is unlimited

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
SM-ALC/SCD-1

5. MONITORING ORGANIZATION REPORT NUMBER(S)
HQ AFLC/MMDAS

6a. NAME OF PERFORMING ORGANIZATION
Sacramento Air Logistics Center

6b. OFFICE SYMBOL (If applicable)
SM-ALC/SCDN

7a. NAME OF MONITORING ORGANIZATION
Wright-Patterson AFB, OH 45433-5000
HQ AFLC/MMDAS

6c. ADDRESS (City, State, and ZIP Code)
SM-ALC/SCDN, McClellan AF Base, CA 95652-5990

7b. ADDRESS (City, State, and ZIP Code)
Wright-Patterson AFB, OH 45433-5000

8a. NAME OF FUNDING/SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State, and ZIP Code)

10. SOURCE OF FUNDING NUMBERS

| PROGRAM ELEMENT NO. | PROJECT NO. | TASK NO. | WORK UNIT ACCESSION NO. |
|---|---|---|---|
| | | | |

11. TITLE (Include Security Classification)
OOPS IT'S HAPPENING

12. PERSONAL AUTHOR(S)
Mel Fisher

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT (Year, Month, Day) | 15. PAGE COUNT |
|---|---|---|---|
| Magazine Article | FROM _____ TO _____ | 31 Aug 1989 | 16 |

16. SUPPLEMENTARY NOTATION
Paper prepared for publication in the SOLE Spectrum magazine as a tutorial on Object Oriented Programming for logisticians.

| 17. COSATI CODES | | |
|---|---|---|
| FIELD | GROUP | SUB-GROUP |
| | | |
| | | |

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Artificial Intelligence, Computers, Application
Object Oriented Programming
Smalltalk (See reverse)

19. ABSTRACT (Continue on reverse if necessary and identify by block number)
Since the introduction of computers, programmers have been searching for higher level languages which support a philosophy of quick and easy application development and maintenance. In 1977, Xerox developed the first object oriented language, Smalltalk. This paper describes the uses and benefits of object oriented programming for the novice.



20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

| 22a. NAME OF RESPONSIBLE INDIVIDUAL | 22b. TELEPHONE (Include Area Code) | 22c. OFFICE SYMBOL |
|---|---|---|
| Roger M. Boan | (513) 257-3201 | HQ AFLC/XPS |

DD Form 1473, JUN 86 — Previous editions are obsolete. — SECURITY CLASSIFICATION OF THIS PAGE

18. Tools, Compuer Programs, Computer Programming, Computer Programs
Programming Languages, Computer Applications

LEAD IN FOR THE ARTICLE.....

INTERESTED IN OOPS? WHAT IS OOPS? THIS TWO PART ARTICLE DISCUSSES THE OBJECT ORIENTED CRAZE THAT IS FOUND ALL OVER THE COMPUTER INDUSTRY AND GETS YOU INTO THE BUSINESS OF UNDERSTANDING THE LATEST OOPS TERMS!

# OOPS IT'S HAPPENING!

By

Mel Fisher

Object Oriented Programming Systems (OOPS) Part I

(This is the first part of an article on OOPS)

# The Software Crisis!

Since the introduction of computers, programmers have been searching for higher level languages which support a philosophy of quick and easy application development and maintenance. Problems in the 50's and 60's resulted in extensions being added to computer languages to make them more modular, thus reducing the problems of having different parts of a program conflict with each other. This module or block of code required protection from other things happening within the program and introduced the concept of an OBJECT and how to protect (encapsulate) data used by that object.

One of the first object oriented languages was Smalltalk, developed at Xerox PARC in 1977 by Alan Kay. It was a tremendous success at influencing the direction of many commercial products like Apple's Lisa and Macintosh. Similar to the Macintosh, the Smalltalk system is not just a language; it's an integrated programming environment. An object oriented programming (OOP) environment like Smalltalk is based on a single universal data structure (the object), a control structure for sending messages, and a uniform class description (the class hierarchy).(KR) These terms will make more sense as we talk about them in part II of this article. For now we just need to know that they are important.

## What good are all these Objects?

The initial appeal of object orientation is that it provides not so much a coding technique as it is a new approach to make software (1) better designed, (2) more reusable, and (3) more reliable.

1) BETTER DESIGNED - OOP provides better concepts and tools to model and represent real world problems by allowing an easier transformation from problem definition to system requirements (user terms) and programming language (computer terms). This transformation from problem to computer specification is often called information modeling, where any real world object can be represented in a variety of ways. For example, we can represent an abstract object (dog) by the table shown below:

DOG

| Dogs name | Breed | Favorite food | Birthdate |
|---|---|---|---|
| Fifi | Poodle | Dry | May 21 82 |
| Rover | Mix | Mix | June 1 83 |

A column in the table represents a characteristic or ATTRIBUTE of the object dog. Each unique dog represents a particular instance of the object dog by a row in the table. Other objects of our real world problem could be:

| Dog Owner | |
|---|---|
| Owner | Address |

| Vet Doctor | |
|---|---|
| Doctor | Address |

The last thing we need to know is that these objects can have relationships and communicate, the owner's name can be associated with the name of a dog. In our case the dog is owned by a owner and can be serviced by a vet doctor. We can begin to create an information model of our problem using these basic concepts to create OBJECTS with ATTRIBUTES which can communicate with other objects through MESSAGES.

2) MORE REUSABLE - Since OOP deals with objects and the data that works with those objects, this packaging of objects (encapsulation) allows for the development of object libraries which can be used when developing applications. These software libraries contain reusable objects (data & methods) that can be incorporated into any program by more than one programmer.

3) MORE RELIABLE - The biggest problem today when designing systems involves an attitude toward change. As application (end user) needs change, the effort involved to make those changes in software is not part of our programming tools, methodologies and concepts. The natural process of using better designed code with reusable software modules will result in systems that adjust to change better and be more reliable.

## So What is Object Oriented Programming?

It is a way for real world problems to be represented as objects, with properties (attributes) about those objects and the operations (messages) permitted to work on object data structures. In an OOP environment, a program obtains information from an object or request an object to do something by sending a MESSAGE to the object. Some objects in any problem will be very similar and thus demonstrate a similar behavior. The object dog above is similar to a Deer. It has four legs, a head, tail, etc., when creating the original object dog it would of made sense to use similar characteristics of the deer object or inherit its characteristics and not reinvent the wheel.

# TRANSITION TO OOPS!

The transition to OOP will not necessarily be a smooth one. There probably will be several phases to go through before programmers can really incorporate OOP into the mainstream of programming.

The first phase of the OOP transition will be to integrate into existing languages OOP features. This incremental process will allow programmers to build on their existing knowledge and experiences and use existing code, incrementally using objects in places where they are needed. This would include languages like C++, Objective C and Object Pascal. Borland and Microsoft recently have announced their versions of Object Pascal while Apple has had Object Pascal for years. Will OOP features be added to other traditional languages (OOP-COBOL) ?

The second phase is to have major application use objects as collections of objects. With this capability, applications can use objects such as a spreadsheet, text or graphics independent of where they were created. These toolkits are just now becoming available for the programmers in the form of Dynamic Link Libraries. At this point, operating systems will have to recognize these collections of objects. Operating systems like OS-2 and Unix offer this in some capacity right now, but it needs to be improved before OOPS makes it to the last phase.

The final or third phase of OOP transition will be tool development for end-users. Programming tools or languages like Smalltalk or Actor provide this now to a certain degree, but some manual coding is still required. Hopefully, tools like this will result in end-users creating applications with object oriented code generators.

## OOP SOURCES


1. Journal of Object-Oriented Programming, 1-800-345-8112.
2. C++ Report, 1-800-345-8112.
3. MacTutor Programming Journal, (714) 640-3730.
4. ParcPlace, Smalltalk-80, 1-800-822-ST80.
5. The Whitewater Group, Actor, (312) 491-2370.
6. Digitalk, Smalltalk/V (pc/Mac) 1-800-922-8255.
7. Yourdon Press, Object-Oriented Systems Analysis by Sally Shlaer and Stephen Mellor.
8. Addison-Wesley, Object Oriented Programming by Brad Cox.
9. Addison-Wesley, "A Little Smalltalk", by Timothy Budd.

# REFERENCES


1. Yourdon Press, "Object-Oriented Systems Analysis" by Sally Shlaer and Stephen Mellor, 1988.
2. Addison-Wesley, "Object Oriented Programming" by Brad Cox, April 1987.
3. PC Week, "OOP: A New Perspective on Code and Data", by Jeffery Duntemann, November 14, 1988.
4. InfoWorld, "Transition to Occur in three Waves", by Stuart J. Johnson, July 3, 1989.
5. Management Information Systems Week, "OOP: more smarts, less code", by David Coursey, June 12, 1989.
6. John Wiley & Sons, "Intelligent Databases", by Kamran Parsaye, Mark Chignell, Setrag Khoshafian, and Harry Wong, 1989.

# OOPS IT'S HAPPENING!

By

Mel Fisher

Object Oriented Programming Systems (OOPS) Part II

(This is part II of a series of articles on OOPS)

## Remember OOPS?

In part I of this article we discussed the software crisis in designing systems with traditional languages, and how Object Oriented Programming (OOP) would provide a new approach for software to be (1) better designed, (2) more reusable and (3) more reliable.

We also talked about how OOP provides a better way to represent real world problems as OBJECTS with properties (ATTRIBUTES) and how objects can communicate with other objects by passing MESSAGES.

## OOP: New ways to Look At Code & Data

Now that we know about objects, attributes, and messages, it is time to add a few more terms to our OOP vocabulary. Under the OOP paradigm, both code and data are important as they are brought together to form an object. Since objects perform actions via methods (procedures), one method might print the object's data on a printer (ie. go print yourself). Another method might request new data for its object which could be a database record object.

This idea of sending messages to objects is intriguing but only part of the story. The most important part of OOP is the idea of INHERITANCE. When an object is defined, it

becomes part of a CLASS of object. In our dog object example in part I, we saw that a dog could have characteristics of a generic animal object. This process is called inheritance. What this means is that any object can inherit the needed characteristics of any other object and then just add the attributes it needs for itself.

Let me state that again because it is the most important part of OOP that makes it dramatically different from any procedural language. The actual inheritance occurs when a new class of object is defined by building upon a previously defined class.

For example: if we create a new object dog, it most likely will have some characteristics of a generic animal object. Since certain animals have four legs, a tail, a head and other features, we can use or inherit these characteristics for our animal object. To make the dog look different, we will add some of our own methods (procedures) to the object to make it unique and behave like a dog.

The last thing to know is that the class of objects is structured in a HIERARCHY. When a new class of an object is created it is considered a descendant of the original class which in turn, is an ancestor to the new class just created. It's similar to a family tree of objects.

Classes higher in the hierarchy tree represent more general characteristics, while classes lower in the hierarchy, represent more specific characteristics common to object classes. For example: the class animal in a hierarchy tree would contain more general characteristics that a dog object which is a certain type of animal.

## A Model For Reality

As mentioned earlier in Part I, the appeal of OOP is that it provides better concepts and tools to model and represent the real world. This allows for a more direct representation of data and modeling of data to the problem at hand.

This process of trying to represent the real world in computer/user terms is called information modeling. Information modeling itself can be very useful in many application areas where there is a need to have systematic processes clarified.

For example, consider a procurement expert within a company. This person has a great deal of knowledge on the inner working of how to get items purchased from various vendors. Information modeling would be helpful here to extract from the procurement specialist, the processes

which make up his/her approach to procurement problems. OOP allows you to represent a model of the procurement problem and the relationships within the problem to assist in the transition from problem definition to computer specifications.

One of the first things you do when building an information model is to define the conceptual units of the problem itself in the form of objects. The combination of all these objects assists in defining the problems scope.

Why is this model important? It represents one of the processes that make up the complete software development lifecycle. This developmental process is typically made up of four phases with a varying degree of each:

1. Analysis of the problem (our model above)
2. Specifications
3. System design
4. Implementation

## Future Directions of OOP?

There are several areas where OOP can be applied; databases, end-user tools, CASE (Computer Aided Software Engineering), and toolkits for programmers. Since the database is the most important part of any information system, this area will benefit the most from incorporating OOP.

A database combined with OOP becomes an intelligent database. To fully utilize intelligent databases, other tools can be added such as expert systems, hypermedia and text management. New information technologies combined with graphics will play an ever-increasing role in information systems of the future.

OOP will not be a cure all for future system design, but it will help reduce the software crisis of today by facilitating software design and maintenance.

# Biography

Mel Fisher is a Knowledge Engineer in the Artificial Intelligence Laboratory (AI Lab) at McClellan AFB, Sacramento, California. He has an A.S. degree in Electronics Technology and a B.S. degree in Instructional Media Technology from California State University, Chico. Prior to joining the AI Lab, most of his experience was in the Aerospace Industry with Lockheed and Martin Marietta. His interests include database systems, object oriented programming, and expert systems.